09/718,228.

WEST Search History

DATE: Monday, August 11, 2003

Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT; PLUR=NO; OP=ADJ		
L21	L20 and (l4 or l5 oor l6) and (l13 or l14 or l15 or l7 or l3)	2	L21
L20	L19 and l2	79	L20
L19	(707/4 OR 707/3).CCLS.	2751	L19
L18	L17 and (l2 or l3 or l11 or l13 or l14 or l15)	10	L18
L17	(l8 or l7) and (l5 or l4) and l1	20	L17
L16	transform$7 same query same specification same XML	1	L16
L15	canonical adj1 format	72	L15
L14	result$1 adj1 transform$3	1207	L14
L13	query adj1 specification	82	L13
L12	L11 near5 l9	0	L12
L11	encapsulat$5	96501	L11
L10	encapsulating	0	L10
L9	query adj1 definition	67	L9
L8	"extensible style sheet language"	4	L8
L7	XSL	100	L7
L6	XSL adj1 transform	5	L6
L5	"extensible markup language"	445	L5
L4	XML adj1 (format or document$1)	238	L4
L3	data adj1 structure	22844	L3
L2	defining near3 query	168	L2
L1	(707/3 OR 707/4 OR 707/100 OR 707/103R OR 715/513).CCLS.	4615	L1

END OF SEARCH HISTORY

Your search matched **[0]** of **[961219]** documents.

You may refine your search by editing the current search expression or entering
a new one the text box. Then click search Again.

xml and xsl and query and encapsulating

Search Again

OR

Use your browser's back button to return to your original search page.

Results:

No documents matched your query.

Your search matched **[0]** of **[959401]** documents.

You may refine your search by editing the current search expression or entering
a new one the text box. Then click search Again.

xml and xsl and query and documents

Search Again

OR

Use your browser's back button to return to your original search page.

Results:

No documents matched your query.

Your search matched **2** of **959401** documents.

A maximum of **2** results are displayed, **25** to a page, sorted by **Relevance** in **descending** order.
You may refine your search by editing the current search expression or entering a new one the text box.

Then click **Search Again**.

xml and xsl and query

Search Again

Results:
Journal or Magazine = **JNL**   Conference = **CNF**   Standard = **STD**

1 **A comprehensive framework for querying and integrating WWW data and services**
Konopnicki, D.; Shmueli, O.;
Cooperative Information Systems, 1999. CoopIS '99. Proceedings. 1999 IFCIS International Conference on , 2-4 Sept. 1999
Page(s): 172 -183

[Abstract] [PDF Full-Text (192 KB)] **IEEE CNF**

2 **An automated client-driven approach to data extraction using an autonomous decentralized architecture**
Blake, M.B.; Liguori, P.;
Autonomous Decentralized Systems, 2001. Proceedings. 5th International Symposium on , 26-28 March 2001
Page(s): 311 -318

[Abstract] [PDF Full-Text (556 KB)] **IEEE CNF**

IEEE Xplore™ ◂ Back to Previous Page

A comprehensive framework for querying and integrating WWW data and services

Konopnicki, D. Shmueli, O.
Dept. of Comput. Sci., Technion-Israel Inst. of Technol., Haifa;

This paper appears in: **Cooperative Information Systems, 1999. CoopIS '99. Proceedings. 1999 IFCIS International Conference on**
Meeting Date: 09/02/1999 -09/04/1999
Publication Date: 1999
Location: Edinburgh , UK
On page(s): 172-183
References Cited: 30
IEEE Catalog Number: PR00384
Number of Pages: xii+361
INSPEC Accession Number: 6359170

Abstract:
Quo is a framework for the development of applications that use WWW data. Quo models the WWW accessible data using a dynamic semistructured data model called Quom (Quasi-Object Model). Within a semistructured model, Quom introduces (I) Object-Oriented features, (2) a structure declaration which is a “lightweight” schema definition. On the other hand, Quom relaxes the requirement, of Object-Oriented data models, of a strict schema definition. This enables dealing with the irregularities of WWW extracted data and with self-describing data formats “a la” XML. Data extraction rules are defined using Quodl, the Quasi-Object Definition Language. Quodl data extraction rules use XSL to analyze and extract HTML/XML data and regular expressions are used for the analysis of and extraction from, raw data

Index Terms:
data models information resources query processing Quo Quodl Quom WWW accessible data WWW data WWW extracted data XSL data extraction rules dynamic semistructured data model extraction rules integrating querying

Documents that cite this document
Select link to view other documents in the database that cite this one.

WEST Search History

DATE: Tuesday, August 12, 2003

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=NO; OP=ADJ			
L11	L10 and (XML or XSL)	0	L11
L10	l2 same tag$1	248	L10
L9	L7 and (query or l1)	3	L9
L8	L7 and query and l1	0	L8
L7	transform$6 same l3	1008	L7
L6	l4 and query	52	L6
L5	L4 and XML	3	L5
L4	l1 and l2	187	L4
L3	lens$2 near3 (load or system)	43843	L3
L2	lens	172431	L2
L1	(707/$ OR 715/$).CCLS.	14649	L1

END OF SEARCH HISTORY

WEST

End of Result Set

Generate Collection Print

L16: Entry 1 of 1 File: USPT Nov 26, 2002

DOCUMENT-IDENTIFIER: US 6487566 B1
TITLE: Transforming documents using pattern matching and a replacement language

Brief Summary Text (23):
In yet another example, in U.S. Pat. No. 4,599,691, Sakaki and Hashimoto describe a tree transformation in machine translation system. XSL (XML Style-sheet Language) is an XML based language specification for rendering XML documents. It has a core tree transformation language. This language is based upon search for an elements that qualify rather than based upon template based patterns as in our system. However, it does not take advantage of the schema structure in the syntax. Even though XSL has syntax to embed scripts for actions on pattern match, it does not integrate a programming language like Java for evaluation conditions of pattern match, conditions of variable evaluation in patterns, and conditions for replacement. There are other XML-based (or otherwise) query languages for XML being proposed that query XML structures and return parts of XML structures that qualify. These are not template based, however, and are not very powerful.

WEST

Generate Collection Print

L18: Entry 6 of 10 File: USPT Jan 14, 2003

DOCUMENT-IDENTIFIER: US 6507856 B1
**** See image for Certificate of Correction ****
TITLE: Dynamic business process automation system using XML documents

Brief Summary Text (10):
In alternate embodiments, the information entered into the template is preferably associated with tag names and the means for merging may include a name tag map for correlating tags names of the template with tag names of the return template. The message may include information having a name and a value and the first parser parses the first message into name and value pairs. The first message may be written in an extensible markup language (XML) and the data type information may be in a corresponding data type definition format (DTD). The means for merging may include a constraint set for identifying tag names used in multiple instances. The constraint set may provide higher level tag names to identify the tag names used in multiple instances. The network is preferably the Internet. The first parser parses the first message to preferably provide tag name and value information in a format of one of a document object model tree and an array.

Brief Summary Text (11):
A system for exchanging and merging extensible markup language (XML) documents over the Internet includes a server accessible by a plurality of remote browsers for transmitting a template including fields for information entry, and a business system accessible by the server for generating a return XML document pursuant to information entered in the template on the browsers. The business system includes a first parser for receiving a first XML document and a corresponding data type definition(DTD)file from a browser, a second parser for receiving a return data type definition(DTD)file to provide a return template and means for merging the first XML document with the return template for providing the return XML document to the browser having portions of the return template with data entered therein corresponding to at least some of the information entered into the template.

Brief Summary Text (12):
In alternate embodiments, the information entered into the template is preferably associated with tag names and the means for merging includes a name tag map for correlating tag names of the template with tag names of the return template. The first XML document may include information having a name and a value and the first parser parses the first XML document into name and value pairs. The means for merging may include a constraint set for identifying tag names used in multiple instances. The constraint set may provide higher level tag names to identify the tag names used in multiple instances. The first parser preferably parses the first XML document to provide tag name and value information in a format of one of a document object model tree and an array.

Drawing Description Text (3):
FIG. 1 depicts a sample XML document encoding a purchase order for use with the present invention;

Drawing Description Text (7):
FIG. 5 is a flow/block diagram of a dynamic XML document exchange system, for business process automation in accordance with the present invention;

Drawing Description Text (9):
FIG. 7 is a flow diagram shown in greater detail of the dynamic XML document exchange

system depicted in FIG. 5 in accordance with the present invention;

Detailed Description Text (2):
The present invention relates to automated document information exchanges and, more particularly, to a system and method for automating document exchange and merging. The document exchange and merge preferably includes the use of extensible Markup Language (XML) documents. An XML name tag map table design, Document Object Model (DOM) tree parsing or serialization, return document template generation, constraint set design, and a document merging algorithm are included in an automated document merging and exchange system, in accordance with the present invention. Although described in terms of XML, DTD and DOM languages/codes, other languages/codes may be implemented in accordance with the invention.

Detailed Description Text (3):
Business documents may be presented by extensible Markup Language (XML) for Internet transmission and World Wide Web access. A business process automation system may receive an XML message or document and its corresponding Data Type Definition (DTD), and generate a return XML message based on a return document DTD, with certain fields pre-filled from the first XML message.

Detailed Description Text (5):
DOM tree parsing or serialization prepares the first document in a suitable data structure, such as tree or array, for efficient processing and matching. The XML parser can be embedded in a web browser such as the Microsoft.RTM. XML parser, or run as a server side application such as IBM Tokyo Research Laboratory XML Parser for Java. The XML parser may receive the first XML document and its DTD, and generate a DOM tree or a serialized name/value pair array. Due to looping, the same tag names may occur multiple times in the DOM tree or the array. Looping includes reusing a format for data entry, for example, a purchase order may include more than one item to be ordered. The same code is used to generate fields for data entry on a template, looping generates the fields needed.

Detailed Description Text (6):
Using parsing techniques, a DTD parser may be created for generating a return document template or a return document DTD parse tree, which can assist the document merge algorithm to prepare the return XML document. The DTD parser transforms the DTD with repeatable and optional fields into a template in tree structure or serialized array with special markers around loop header nodes or name tags. Optional fields may include a second business address or phone number, for example.

Detailed Description Text (8):
A document merging algorithm, in accordance with the invention, generates a return XML document, by either sequentially scanning the name tags from the template in an array structure, or recursively traversing the DTD tree node from the template in a tree structure, to match their counterparts in the XML DOM tree or the serialized array using the XML name tag map table. If a match is found, the corresponding value of the first XML message is retrieved and treated as the value associated with the current name tag. When a name tag with the special marker is detected, a loop is found or revisited, in which case the loop header tag is checked with the constraint set for loop count, or the tags inside the loop are matched against the XML DOM tree or serialized array, to determine if the content of the loop should be generated again. The algorithm handles both variable number and fixed number of loop iterations. Referring now to the drawings in which like numerals represent the same or similar elements and initially to FIG. 1, a sample XML document is depicted for encoding a Purchase Order (PO), where there are two items ordered, i.e., item Nos. 0001 and 0002. Each item includes information code which begins at line items (LineItem) 20 and (LineItem) 25, respectively. Line item 20 includes detailed product descriptions, service types, and ship to address, and the line item 25 includes only key information, such as price, quantity and unit. Other information is included, for example, address information 21 and a total amount of the purchase 23.

Detailed Description Text (9):
Referring to FIG. 2, a Data Type Definition (DTD) for the sample PO of FIG. 1 is shown. DTD defines 26 data elements (0-25). The Address data element 6 is referred by the data elements Purchase Order 0 and LineItem 5, where LineItem itself is referred

by PurchaseOrder 0 as indicated by the numeral 30. Repeatable data elements are marked with a "*", e.g., the LineItem indicated by 30 referred by PurchaseOrder 0, and optional data elements which can occur zero or once are marked with a "?", e.g., AdditionalName indicated at numeral 32 referred by Address 6. The "#PCDATA" 34 represents parsed character data. The style sheet written in, for example, Javascript, XSL, or CSS provides a way to render the XML document (FIG. 1) to a browser (see FIG. 4).

Detailed Description Text (14):
In the illustrative example shown in FIG. 6, a buyer runs a web server 206. In step 201, a supplier can visit the buyer's web site to view PO's using a standard web browser 207. The supplier may decide to create a corresponding invoice from the received PO by submitting a "prepare invoice" request 202 to the web server. The XML document exchange/merge system 105 on the buyer side is invoked and dynamically generates a partial invoice 202'. The partial invoice in XML format is transmitted over the Internet and displayed on supplier's browser 203. The supplier can edit the partial invoice 204, and submit the completed invoice back to the buyer for record handling or auditing 205. The system 105 can also be run on the supplier side browser as programs written in Javascript, or as Java applets, for example.

Detailed Description Text (15):
Referring to FIG. 7, an internal flow diagram of a dynamic XML document exchange system 105 is shown. A standard XML parser 305 takes the input XML 125 and DTD 115, and generates an intermediate structure, a tree 355 or an array 355', which serves as part of the input data to a merge algorithm 335. The XML parser 305 may be a client side application, which may serialize tree elements into an array of hyper-text markup language (HTML) components 355', or a server side stand-alone application, which may construct the tree structure 355 (See FIGS. 10A and 10B). After parsing the return document DTD 135, the DTD parser 315 creates a template 365 in either array format 605 or tree structure 615, as shown in FIGS. 10A and 10B, respectively.

Detailed Description Text (24):
The first XML document may include more than the X iterations or map items 730 needed to fill in the return document template, these items may be skipped and not appear in the return document in block 740.

Detailed Description Text (29):
Having described preferred embodiments of a dynamic business process automation system using XML documents (which are intended to be illustrative and not limiting), it is noted that modifications and variations can be made by persons skilled in the art in light of the above teachings. It is therefore to be understood that changes may be made in the particular embodiments of the invention disclosed which are within the scope and spirit of the invention as outlined by the appended claims. Having thus described the invention with the details and particularity required by the patent laws, what is claimed and desired protected by Letters Patent is set forth in the appended claims.

Current US Original Classification (1):
715/513

Current US Cross Reference Classification (5):
707/3

CLAIMS:

3. The system as recited in claim 1 wherein the first message is written in an extensible markup language (XML) and the data type information is included in a corresponding data type definition format (DTD).

8. A system for exchanging and merging extensible makeup language (XML) documents over an Internet comprising: a server accessible by a plurality of remote browsers for transmitting a template including fields for information entry; and a business system accessible by the server for generating a return XML document pursuant to information entered in the template on the browsers, the business system including: a first parser for receiving a-first XML document and a corresponding data type definition (DTD) file

from a browser, the first XML document including name tags; a second parser for receiving a return data type definition (DTD) file to provide a return template, the return template including name tags; and means for merging the first XML document with the return template for providing the return XML document to the browser, the return XML document having portions of the return template with data entered therein corresponding to at least some of the information entered into the first XML document, wherein the means for merging includes a name tag map for correlating the name tags of the first XML document with the name tags of the return template.

9. The system as recited in claim 8 wherein the first XML document includes information having a name and a value and the first parser parses the first XML document into name and value pairs.

12. The system as recited in claim 8 wherein the first parser parses the first XML document to provide tag name and value information in a format of one of a document object model tree and an array.

WEST

Generate Collection Print

L18: Entry 1 of 10 File: USPT Jun 17, 2003

DOCUMENT-IDENTIFIER: US 6581062 B1
TITLE: Method and apparatus for storing semi-structured data in a structured manner

Brief Summary Text (5):
Increasingly, because of its richness in functions and extensibility, information pages, such as web pages, are being constructed using the extensible style language (XSL) and semi-structured data, such as extensible markup language (XML) encoded data.

Brief Summary Text (6):
"Semi-structured data" refers to data that has structure, but where the contents of particular structural elements need not be consistent. To facilitate this characteristic, data are "self-describing". For example, in a "person" application, a person can be validly defined by semi-structured data with only a subset of all possible data associated with a person, e.g., by only a last name and a telephone number, or a first name, last name, and address, or some other combinations. Or, a person may be defined with additional data not previously seen, such as an employer name, an employer address, and an employer telephone number. Thus, each semi-structured "person" definition may vary.

Brief Summary Text (10):
One significant issue, however, is how to convert from semi-structured data, such as XML encoded data, to structured data storage, such as a SQL database. Towards this end, various approaches have been proposed. For example, see Florescu et al., A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Database, Rapport de Recherche No. 3680, INRIA, Rocquencourt, France (May 1999), discusses techniques and issues related to using directed graphs to represent semi-structured data. And, Shanmugasundaram et al, Relational Databases for Querying XML documents: Limitations and Opportunities, Proceedings of the 25.sup.th VLDB Conference, Edinburgh, Scotland (1999), discusses using XML document type descriptors (DTDs) to convert XML documents to relational tuples.

Detailed Description Text (21):
FIG. 10 illustrates an overview of yet another aspect of the present invention. Shown are information pages 302 formed using a language that allows for the use of semi-structured queries, such as XSL, and employment of semi-structured data, like XML encoded data, stored in a structured manner. The correspondence between the semi-structured data and the structured organization are described using the earlier described meta-table or a description mechanism of like kind. Pre-processor 304, incorporated with the teachings of the present invention, is equipped to pre-compile information pages 302, to generate pre-processed information pages 302', replacing the semi-structured queries with equivalent structured queries to retrieve the required data from the structured data storage. Thus, at fulfillment time, that is in response to a request for one of the information pages, the requested information page may be dynamically completed with the required data, without having to determine in real time where the required semi-structured data are stored in the structured data storage. As a result, a request may be fulfilled with a shorter latency. In other words, the present invention also advantageously enables speed up of fulfillment of requested information pages that have to be dynamically completed with semi-structured data retrieved in real time.

Detailed Description Text (23):
Thus, pre-processed information pages 302' are now primed to readily respond to their

requests. FIGS. 12a-12c illustrate a specific example of pre-processing an information page. Shown in FIG. 12a is an example XSL document 502 having a number of match templates. Each matching template includes one or more data extraction commands, such as select, value-of, apply template, and the like. FIG. 12b illustrates a schema of the underlying semi-structured data 504. FIG. 12c illustrates the resulting replacement structured query ("Query Loop") 506, including the join conditions, and the control structure to re-use the structured query ($QL.1, $QL.2, and so forth).

Detailed Description Text (25):
FIG. 14 illustrates an example network environment suitable for exploiting information pages pre-processed in accordance with the present invention. Network environment 700 includes web server 702 and a number of client computers 704 coupled to web server 702 through network 706. Web server 702 is provided with information pages formed with a language like XSL, using semi-structured data, like XML encoded data, stored in a structured data storage, and the information pages are pre-processed as earlier described, i.e. with the semi-structured queries being replaced by equivalent structured queries (and associated control structures, if any). Client computers 704 request selected ones of the information pages from web server 702. Web server 702 fulfills the requests, dynamically completing the information pages, retrieving the required data from the structured data storage, using the replacement equivalent structured queries. Accordingly, the requests of client computers 704 are fulfilled with shorter latencies.

Current US Original Classification (1):
707/100

Current US Cross Reference Classification (3):
707/103R

Current US Cross Reference Classification (5):
715/513

Other Reference Publication (1):
Tufte et al., Relational Databases for Querying XML Documents, Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999.*

Other Reference Publication (3):
Tufte et al., Relational Databases for Querying XML Documents, Proccedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999.*

Other Reference Publication (5):
Shanmugasundaram et al., "Relational Databases for Querying XML Documents:Limitations and Opportunities". In Proc. of the 25th VLDB Conf., Edinburgh, Scotland, 1999, 13 pgs.

CLAIMS:

1. A method in a computer system for converting semi-structured data to structured data, the method comprising: providing a mapping data structure that maps elements of the semi-structured data to elements of the structured data; generating a structured schema for the structured data based on the provided mapping data structure; and storing the semi-structured data as structured data in accordance with the generated structured schema wherein different mapping data structures can be provided to convert semi-structured data to structured data in accordance with different generated structured schemas.

2. The method of claim 1 wherein the provided mapping data structure is generated automatically from a semi-structured schema of the semi-structured data.

4. The method of claim 1 wherein the mapping data structure is generated manually.

7. The method of claim 1 including: retrieving stored structured data; and storing the retrieved structured data as semi-structured data in accordance with the provided mapping data structure.

8. The method of claim 1 wherein the provided mapping data structure includes for each parent and child relationship in the semi-structured data, a mapping from a parent identifier and a child identifier of the parent and child relationship to a table identifier of a table in the structured data that corresponds to the parent and child relationship.

16. A method in a computer system for converting data stored in an XML format into data stored in a relational format, the method comprising: providing a mapping description between the data in the relational format and the data in the XML format wherein for each parent and child relationship in the XML data, the provided mapping description includes a mapping from a parent identifier and a child identifier of the parent and child relationship to a table identifier of a table defined by the relational schema that corresponds to the parent and child relationship; generating a relational schema for the data in the relational format based on the provided mapping description; and storing the data in the XML format as data in the relational format in accordance with the generated relational schema and the provided mapping description.

17. The method of claim 16 wherein the provided mapping description is generated automatically from an XML schema for the data in the XML format.

18. The method of claim 17 wherein the XML schema is derived from analysis of the data in the XML format.

20. The method of claim 16 including: retrieving data stored in the relational format; and storing the retrieved data in the XML format in accordance with the provided mapping description.

23. The method of claim 16 wherein the mapping supports recursive definition of elements of the data in XML format.

28. A method in a computer system for converting structured data to semi-structured data, the method comprising: providing a mapping data structure that maps elements of the structured data and elements of the semi-structured data; and generating a semi-structured schema for the semi-structured data based on the provided mapping data structure; storing structured data as semi-structured data in accordance with the generated semi-structured schema wherein different mapping data structures can be provided to convert the structured data to semi-structured data in accordance with different generated semi-structured schemas.

29. The method of claim 28 wherein the provided mapping data structure is generated automatically from a structured schema for the structured data.

31. The method of claim 28 wherein the provided mapping data structure is generated by a user.

33. The method of claim 28 wherein the generated semi-structured schema defines an XML format.

34. The method of claim 28 including: retrieving stored semi-structured data; and storing the retrieved semi-structured data as structured data in accordance with the provided mapping data structure.

35. The method of claim 28 wherein the provided mapping data structure includes for each parent and child relationship of the semi-structured schema, a mapping from a parent identifier and a child identifier of the parent and child relationship to a table identifier of a table in the structured data that corresponds to the parent and child relationship.

43. A method in a computer system for converting data stored in a relational format into data stored in an XML format, the method comprising: providing a mapping description that maps the data in the relational format to the data in the XML format wherein for each parent and child relationship in the XML data, the generated mapping description includes a mapping from a parent identifier and a child identifier of the parent and child relationship to a table identifier of a table defined by the

relational schema that corresponds to the parent and child relationship; generating an XML schema for the data in the XML format based on the provided mapping description; and storing the data in the relational format as data in the XML format in accordance with the generated XML schema and the generated mapping description.

47. The method of claim 43 including: retrieving data stored in the XML format; and storing the retrieved data in the relational format in accordance with the generated mapping description.

60. The computer system of claim 55 wherein the semi-structured data is in an XML format.

64. A method in a computer system for converting data between a semi-structured format and a structured format, the method comprising: providing a mapping data structure that maps elements of the structured format and elements of the semi-structured format; retrieving data in one of the formats; and storing the retrieved data in the other format in accordance with the provided mapping data structure wherein different mapping data structures can be used to convert data having one schema to data having another schema.

66. The method of claim 64 wherein the semi-structured format is an XML format.

74. The method of claim 70 wherein the semi-structured format is an XML format.

WEST

Generate Collection | Print

L21: Entry 1 of 2 File: USPT Aug 5, 2003

DOCUMENT-IDENTIFIER: US 6604100 B1
TITLE: Method for converting relational data into a structured document

Brief Summary Text (4):
XML (eXtensible Markup Language) can serve many purposes. XML is a more expressive markup language than HTML (Hyper-Text Markup Language). XML may be an object-serialization format for distributed object applications. XML serves as the standard format for data exchange between inter-enterprise applications on the Internet. In data exchange, XML documents are generated from persistent data and then sent over a network to an application. To facilitate data exchange, numerous industry groups, such as healthcare and telecommunications groups, have been defining public document type definitions (DTDs) that specify the format of the XML data to be exchanged between their applications. The aim is to use XML as a "lingua franca" for data exchange between inter-enterprise applications. XML could make it possible for data to be exchanged regardless of the platform on which it is stored or the data model in which it is represented.

Brief Summary Text (6):
Relational data is tabular, flat, normalized, and its schema is proprietary, which makes it unsuitable for direct exchange. In contrast, XML data is nested and un-normalized, and its DTD is public. Thus, the mapping from relational data to an XML view is often complex, and a conversion tool must be general enough to express complex mappings. Existing commercial systems fail to be general, because they map each relational database schema into a fixed, canonical DTD. This approach is limited, because no public DTD will match exactly a proprietary relational schema. In addition, it is often desirable to map one relational source into multiple XML documents, each of which conforms to a different DTD. Hence a second step is required to transform the data from its canonical form in XML into its final XML form.

Brief Summary Text (7):
Also, the tools must be dynamic, i.e., only the fragment of the XML document needed by the application should be materialized. In database terminology, the XML view must be virtual. The application typically specifies in a query what data item(s) it needs from the XML document. Typically, these items are a small fraction of the entire data. Some commercial products allow users to export relational data into XML by writing scripts. However, these tools are not dynamic. Rather, they are general because the entire document is generated all at once.

Brief Summary Text (9):
Several commercial tools for exporting relational data into XML views exist today. The ODBC2XML, a product of Intelligent Systems Research (www.intsysr.com) tool allows users to define XML documents with embedded SQL statements, which permit the users to construct an XML view of the relational data. Such views are materialized, however, and cannot be further queried with an XML query language. Alternatively, Oracle's XSQL tool defines a fixed, canonical mapping of the relational data into an XML document, by mapping each relation and attribute name to an XML tag and tuples as nested elements. Such a view could be kept virtual, but this approach is not general enough to support mapping into an arbitrary XML format. IBM's DB2, XML Extender provides a Data Access Definition (DAD) language that supports both composition of relational data in XML and decomposition of XML data into relational tables. DAD's composition feature supports generation of arbitrary XML from relational data. However, the criteria for grouping elements is implicit in the DAD and DAD specifications cannot be nested arbitrarily. More significantly, XML Extender does not support query

composition.

Detailed Description Text (13):
Typically, applications contact SilkRoute 100, to request data. An application 120, only "sees " the virtual XML view, not the underlying relational database. To access the data, the application 120, can formulate a user query in XML-QL over the virtual view and send it to SilkRoute 100. Together, the view query (e.g., RXL view query) and the user query (e.g., XML-QL user query) can be passed to the query composer module 102, in SilkRoute 100. The query-composer module 102, computes the composition and produces a new view query (e.g., RXL query), called the executable query. The answer to the executable query typically includes only a small fragment of the database, e.g., one data item, a small set of data items, or an aggregate value. The result of SilkRoute 100, is an XML document, as specified by the user query (e.g., XML-QL user query).

Detailed Description Text (15):
Until now, SilkRoute 100, has manipulated only query source code, but no data. At this point, the data extraction part (e.g., SQL queries) is sent to the RDBMS server 110, which returns one tuple stream per each query (e.g., SQL query) in the data extraction part. The XML generator module 106, merges the tuple streams with the XML-construction part and produces the XML document, which is then returned to the application 120.

Detailed Description Text (16):
This scenario is probably the most common use of SilkRoute. However those skilled in the art will recognize that minor changes to the information flow in FIG. 1 can permit other scenarios. For example, the data administrator may export the entire database as one large XML document by materializing the view query. This can be done by passing the view query directly to the translator. In another scenario, the result of query composition could be kept virtual for later composition with other user queries. This is useful, for example, when one wants to define a new XML view from an existing composed view.

Detailed Description Text (28):
When Skolem terms are missing, RXL introduces them automatically. Since Skolem terms could be used to define arbitrary graphs, RXL enforces semantic constraints that guarantee a view always defines a tree, and therefore, a well-formed XML document. For example, the Skolem term of a sub-element must include all the variables of its the parent element.

Detailed Description Text (39):
The where clause includes a pattern (lines 3-10) and a filter (line 11). A pattern's syntax is similar to that of XML data, but also may contain variables, whose names start with $. Filters are similar to RXL (and SQL). The meaning of a query is as follows. First, all variables in the where clause are bound in all possible ways to the contents of elements in the XML document. For each such binding, the construct clause constructs an XML value. Grouping is expressed by Skolem terms in the construct clause. In this example, the construct clause produces one result element for each value of $company. Each result element contains the supplier's name and a list of name elements containing the product names.

Detailed Description Text (47):
The translator 104, also takes a source description, which is an XML document specifying systems information needed to contact the source: the protocol (e.g. JDBC), the connection string, and a source-specific query driver. The driver translates RXL expressions into the source's query language, which is typically a dialect of SQL. Although one skilled in the art will appreciate that other query languages can be supported. For example, the executable RXL query (C) is translated into the following SQL query: select c.pid as pid, c.item as item from Clothing c, SalePrice s where c.category="outerwear", c.pid=s.pid, s.price<0.5*c.retail sort by c.pid

Detailed Description Text (50):
In this example, the translation requires only one SQL query. In general, there may be several ways to translate a complex RXL query into one or more SQL queries and to merge tuple streams into the XML result. Choosing an efficient evaluation strategy may be important when the RXL query returns a large result, e.g., if the entire XML view

is materialized. SilkRoute can have one or more evaluation strategies, which can generate one SQL query for each disjunct of an RXL sub-query, which must be in disjunctive-normal form (DNF). Each SQL query has a sort by clause, making it possible for the XML generator 106, to merg the queries into an XML document in a single pass.

Detailed Description Text (57):
In this section, the query composition algorithm is described. As discussed previously, an RXL query, such as V, takes a relational database as an input and returns an XML document as an output. The XML-QL user query, such as U, which is written against V, takes an XML document as an input and returns an XML document. For any database D, the result of U can be computed by first materializing V(D), denoted as XMLD, and then computing U(XMLD). The query composition problem is to construct an equivalent RXL query C, where C=U V. In other words, it would be desirable to construct an RXL query C that is guaranteed to yield the same result as U and V for any database D, that is, C(D)=U(V(D)). C takes as an input a relational database and returns an XML document. With C, the construction of the intermediate result XMLD is skipped. As an example, RXL view query (V), and XML-QL user query (U) can be used with the result of the composition, C, being composed RXL query (C).

Detailed Description Text (67):
For the composition algorithm, the view query V may be represented by a data structure called a view tree, which includes a global template and a set of datalog rules. The global template can be obtained by merging all view query (V) templates from all its construct clauses. Nodes from two different templates may be merged if and only if they have the same Skolem function. Hence, each Skolem function occurs exactly once in the view tree. The datalog rules are non-recursive. Their heads are the Skolem functions names, and their bodies include relation names and filters. The datalog rules can be constructed as follows. For each occurrence of a Skolem function F in a view query (V), one rule is constructed of the form F(x, y, . . .):-body, where body is the conjunction of all from and where clauses in the scope where F occurs. When a rule is associated with a Skolem function, then that rule guards the Skolem function and its corresponding XML element. In both the template and datalog rules, the tuple variables used in RXL can be replaced by column variables.

Current US Original Classification (1):
707/3

Other Reference Publication (1):
Jayavel Shanmugasundaram, H. Gang, Kristin Tufte, Chun Zhang, David DeWitt, and Jeffrey F. Naughton. Relational Databases for Querying XML Documents: Limitations and Opportunities. In Proceedings of the Conference on Very Large Data Bases, 1999.*

Other Reference Publication (8):
J. Shanmugasundaram, E. Shekita, R. Barr, M. Carey, B. Lindsay, H. Pirahesh, B. Reinwald, "Efficiently Publishing Relational Data as XML Documents", VLDB Journal (to appear).

CLAIMS:

41. A method of forming a composed query for use in converting relational data of a relational database into XML data, said method comprising the steps of: receiving a user query in an XML based language requesting relational data; forming a view query in a transformation language, the view query defining a structured document view of the relational data and having a structure that is independent from a structure of the relational data of the relational database; and composing an executable query in the transformation language from the view query and the user query, wherein the structured document view is capable of defining a document of arbitrary nesting depth.